December 1987

UILU-ENG-87-2271
CSG-75

COORDINATED SCIENCE LABORATORY
College of Engineering

LANGLEY GRANT
IN-61-CR
111917
p. 45

DIAGNOSTIC REASONING IN DIGITAL SYSTEMS

Kurt Henry Thearling

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Approved for Public Release. Distribution Unlimited.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-87-2271 (CSG 75)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL *(If applicable)* N/A	7a. NAME OF MONITORING ORGANIZATION NASA
6c. ADDRESS *(City, State, and ZIP Code)* 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS *(City, State, and ZIP Code)* NASA Langley Research Center Hampton, VA 23665
8a. NAME OF FUNDING/SPONSORING ORGANIZATION NASA 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER NASA-NAG-1-613
8c. ADDRESS *(City, State, and ZIP Code)* See 7b.	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*

Diagnostic Reasoning in Digital Systems

12. PERSONAL AUTHOR(S)

Thearling, Kurt Henry

13a. TYPE OF REPORT Technical	13b. TIME COVERED FROM ______ TO ______	14. DATE OF REPORT *(Year, Month, Day)* December 1987	15. PAGE COUNT 41

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*

fault diagnosis, artificial intelligence, techniques, reasoning, partitioning

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Described in this thesis is an efficient method for fault diagnosis in digital systems based on the technique of reasoning. The methodology operates on the observed erroneous behavior and the structure of the system. The behavior consists of the error(s) observed on the circuit's output lines and specific values on the circuit's input lines. The techniques described in this thesis improve upon previously published research on diagnostic reasoning in two ways. Previous work has stressed system independent techniques which could be used to diagnose any faulty system whose structure can be represented. By concentrating on the specific case of diagnosing faulty digital circuits, it is possible to simplify the representation of the structure of the system. This representation, in the form over

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE *(Include Area Code)* 22c. OFFICE SYMBOL

83 APR edition may be used until exhausted.
All other editions are obsolete.

of an AND/OR fault tree, efficiently abstracts the structure of a faulty digital system. More importantly, a method for partitioning the digital system is introduced which can considerably reduce the runtime complexity of a diagnosis.

DIAGNOSTIC REASONING IN DIGITAL SYSTEMS

BY

KURT HENRY THEARLING

B.S.E., University of Michigan, 1985
B.S.E., University of Michigan, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1988

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

NOVEMBER 1987

WE HEREBY RECOMMEND THAT THE THESIS BY

KURT HENRY THEARLING

ENTITLED DIAGNOSTIC REASONING IN DIGITAL SYSTEMS

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF MASTER OF SCIENCE

R. K.

Director of Thesis Research

Head of Department

Committee on Final Examination†

Chairperson

† Required for doctor's degree but not for master's.

O-517

ABSTRACT

Described in this thesis is an efficient method for fault diagnosis in digital systems based on the technique of reasoning. The methodology operates on the observed erroneous behavior and the structure of the system. The behavior consists of the error(s) observed on the circuit's output lines and specific values on the circuit's input lines. The techniques described in this thesis improve upon previously published research on diagnostic reasoning in two ways. Previous work has stressed system independent techniques which could be used to diagnose any faulty system whose structure can be represented. By concentrating on the specific case of diagnosing faulty digital circuits, it is possible to simplify the representation of the structure of the system. This representation, in the form of an AND/OR fault tree, efficiently abstracts the structure of a faulty digital system. More importantly, a method for partitioning the digital system is introduced which can considerably reduce the runtime complexity of a diagnosis.

ACKNOWLEDGEMENTS

I would like to thank Professor Ravi Iyer for his encouragement and assistance during the development of this work. A special thanks is also owed to my friends in the Computer Systems Group. Those late night "hen runs" were just the right thing to clear one's mind and provide a necessary distraction. In addition, I would like to thank my family for their love and encouragement during my studies.

This work was supported by the National Aeronautics and Space Administration under contract number NAG-1-613.

TABLE OF CONTENTS

1. INTRODUCTION

In the area of reliable computation, the concept of fault diagnosis is a central issue. Once an error is observed, the faulty component responsible for the error must be located and replaced (either by a system technician or by automated spare reconfiguration). By examining the error and the structure of the system, possible sources of the error can be determined. The development of a methodology which determines these possible sources is the aim of this thesis.

In a digital circuit, there is a definite flow of signals from the inputs to the outputs. These signals (which take on a binary set of values) flow through system components that modify the signal values according to a functional specification. When one of the components is faulty, the value that a particular signal takes on can become erroneous. One approach to fault diagnosis is to determine the relationship between the components and the way they can, if faulty, affect the signal values.

Presented in this thesis is an approach to fault diagnosis based on the concept of reasoning. Reasoning can be used to produce a methodology which is both natural and efficient. The structures used to describe the relationship between the digital system components and the signal values within the system can be easily represented with first-order predicate calculus and clause-form theorems. The reasoning required to determine possible sources of any observed errors can be performed on these structures using simple logical transformations. The process of diagnosing a digital system takes information about the erroneous activity of the system, the inputs applied to the system, and the structure of the system and generates a set of suspect components. A suspect is any system component which could have produced (under fault) the observed error activity, given the known inputs to the system. All components which could have produced the error activity are included in the suspect set. This guarantees that the actual faulty component will be included in the suspect set. The assumption is made that there is only a single faulty component present in the system.

The contributions of this thesis are in two areas. First, unlike other methods, an explicit AND/OR fault tree representation is used to represent the circuit structure. This representation reduces the search complexity for the digital circuit diagnosis while keeping a structure isomorphic to the circuit. The fault tree is traversed for the purpose of a diagnosis and is transformed into logical representations of the signal flows which are referred to as theorems. The theorems are then used to determine the faulty component responsible for observed error(s).

Another important contribution of this thesis is the introduction of a partitioning technique which reduces the overall complexity of a diagnosis. A major disadvantage of past approaches involving diagnostic reasoning in digital systems is in the complexity of the diagnosis once an error is observed. A new technique to partition the circuit such that each partition has its own AND/OR fault tree and theorem representation is introduced. Each partition is diagnosed separately and only those partitions whose components could have produced the error are examined. By preprocessing each partition during the design phase of the circuit, it is shown that the complexity of the runtime diagnosis can be reduced. Since the time between the observance of an error and the determination of the sources is critical, much of the diagnostic process can be done in advance.

2. RELATED RESEARCH

Previous work in the area of digital system diagnosis has come from both AI and non-AI based approaches. Most of the non-AI related work in this area has used the PMC (Preparata, Metze, and Chien) model for digital system diagnosis [1]. This model is aimed at multiple processor systems with a directed graph used to describe the connections between processors. An assumption is made which limits the number of faulty processors to T processors. A system being diagnosed under such an assumption is referred to as a T-fault diagnosable system. The basic concepts of diagnosing systems described by this model are as follows. The system is first partitioned into separate processors, with each processor testing some subset of the other processors. The results of the tests can take on two possible values: the processor under test is faulty or it is fault-free. These test results are assumed to be correct. The results of all the tests are referred to as a syndrome. By looking at the graphical representation of the system (in the form of a connection matrix) and the syndrome, a diagnosis is performed. In [1], various bounds on the number of processors required to perform a T-fault diagnosis are developed. An efficient polynomial time algorithm to perform a diagnosis based on this model has been developed by Dahbura and Masson [2].

The PMC model is not readily applicable to diagnosing the circuits of the type considered here. The PMC model is at the system level while we are considering the gate/module level; our model is essentially one level lower than the PMC model in the design hierarchy of a digital system.

In the area of artificial intelligence, the diagnosis of a faulty digital circuit falls within the area of reasoning about physical systems [3-5]. There are two approaches to solving this problem. Both can be applied to the diagnosis of digital systems. The first approach is to use what is commonly known as a "rule-based" expert system. Instead of developing general reasoning methodologies suitable for the diagnosis of any digital system, rule-based systems are driven by a set of device specific rules developed specifically for a particular system. The rules are generated by human or computer "experts" knowledgeable about the system behavior. The rules are based on knowledge of the operation of the system, previous failure data, and possibly a causal analysis of the system similar to the reasoning approach (which is described later). By generating a sufficient number of these rules, it it possible to describe the behavior of a system. There have been a large number of diagnostic expert systems developed using this approach [6-16]. These diagnosis rules are typically of the form *If* < X is true > *Then* < add Y to the suspect list >. An example, from [9], is

IF (1) the instrument is outputting zero or low voltage, AND
   (2) the condition of component F1 is not nominal

THEN the fuse is definitely faulty

This rule would "fire" if conditions (1) and (2) were found to be true. If this were to happen, the component listed in the "then" statement would be added to the set of suspect components.

As would be expected, a rule-based system requires a large number of rules to adequately cover the most common faults. As an example, the diagnosis system in [9] has 356 different rules. But even with a large number of these rules, it is virtually impossible to guarantee that all faults can be diagnosed correctly. In addition, the generation of these rules can be a very time consuming process. Whenever there is a change to the structure of the system, the rules must be modified accordingly by human experts knowledgeable about the operation of the system.

The second artificial intelligence approach to diagnosis is to apply reasoning techniques to obtain an understanding of the behavior of a system. By applying the knowledge of simple component behavior, the behavior of an entire system composed of these components can be determined. In contrast to the rule-based diagnosis systems, diagnostic reasoning attempts to take simple component behavior and understand the behavior of an entire system. This allows flexibility, in that the system can diagnose any circuit composed of components which it "understands." The major contributions to this area have been made by Davis [17-21] and Genesereth [22,23].

Davis's work has stressed the development of a representation suitable for the description of the system structure and component behavior. Once a suitable representation has been obtained, a technique known as constraint suspension [20] is used to perform a diagnosis. The procedure starts with predicting (through simulation) the output value given the input values and certain constraints which describe the behavior of the system components. When the predicted value is different from an observed value (which is obtained by applying input values to the actual circuit and monitoring the output value), the system is faulty and a diagnosis is carried out. To first determine the possible sources of the error, the components between the erroneous output and an input are listed as "candidates." For each one of the candidates, a simulation is performed. This simulation is different from the initial simulation in that the constraints that describe the behavior of a component are suspended. The diagnosis procedure modifies the behavior of each of the candidates (in separate simulations) and checks to see if this modified behavior could have produced the observed error. If a modified component is capable of producing the observed error, it is added to the list of suspects. This procedure is repeated for every observed error.

The method most closely related to the work described in this thesis has been developed by Genesereth. In his diagnosis methodology, a set of behavioral descriptions is developed to describe the behavior of system components. The structure of the system is represented in the form of connections between instances of these components. A backward searching procedure (depth first search) is used to search the structure of the system using behavioral descriptions of the components. The search begins at the output which was observed to produce an error. At each node (component) in the search, the list of behavioral descriptions is searched to find those whose behavior is consistent with the observed error. When the search reaches a system input, the input is compared with the value expected to be on that line (given the error value on the output). If the values are different, the components on the path from that input to the output are added to the list of potential sources of the error. A pruning of the diagnostic search space is done at runtime by removing portions of the search space unreachable due to known input values. This allows an improved average case time complexity but does not improve the worst case time complexity. The worst case occurs when the application of input values (while traversing the search space) does not prune any of the search space (i.e., all inputs are at the edge of the search space). This process of searching the structure is repeated for every error that is observed. As the process is repeated for additional errors, it is possible to reduce the number of suspect components.

There are a number of areas within the previous research that need to be improved upon. These areas are addressed in the subsequent sections of this thesis. First, the relationship between the structure of the system and the flow of errors through the structure is not explicitly described. This relationship is the essence of a diagnosis in a digital system. By embodying this concept in the diagnosis data structures, it is possible to reduce the complexity of the diagnosis while simplifying its representation. Another problem with the previous work is the lack of techniques to deal with the complexity of the runtime diagnosis. A solution to alleviate these problems can greatly aid in the the implementation of diagnostic reasoning techniques for real world applications.

3. OVERVIEW OF THE DIAGNOSIS METHODOLOGY

For a diagnosis to take place, certain information must be available. This information includes the observed error, the inputs responsible for the generation of the error, and the structure of the system being diagnosed. Once this information is available, a diagnosis can be performed.

The proposed diagnosis procedure is broken up into two steps. In the first step, the structure of the system is traversed to produce theorems that represent information about the possible sources of errors in the system. This step is done in the form of preprocessing to reduce the time required to perform the diagnosis once an error is observed. The structure of the circuit is assumed to be in the form of a netlist or a similar representation. This structure is abstracted to create an AND/OR fault tree, which differs somewhat from the well-known fault tree representation [24] in that it is slightly simpler. The simplification is due to the well-defined faulty and nonfaulty behavior of the digital system components. For each of the errors that can be observed on the circuit outputs, an AND/OR fault tree for the circuit is derived. To reduce the complexity of the diagnostic process, the circuit is first partitioned before it is traversed. Instead of traversing the entire circuit, each partition is traversed separately. By appropriately choosing the partitions, a considerable reduction in the runtime complexity of the diagnosis can be achieved.

When the circuit is in operation and an error is observed, information about the error and inputs are applied to the theorems generated above to determine the suspect component(s). The suspect set can be further refined if additional errors are observed for the same fault with different input values. Note that the first step (generating the theorems) is only performed once and is *not* repeated for additional diagnoses. This step needs to be done only once. Assuming that there is only a single fault, the suspect component must exist in the suspect sets for all diagnoses. Thus, an intersection of the suspect sets allows a refinement of the diagnosis.

4. MODELS AND DEFINITIONS

The systems of interest are assumed to be combinational circuits. They are composed of modules which can be as simple as a gate (OR, AND, etc.) or more complex (such as adders, multiplexors, etc.). The fault model under consideration is the single module fault. This model is considerably more comprehensive than the classical stuck-at model. A fault is defined to be any number of changes in the truth table representation of a module's behavior which does not cause the module to exhibit sequential behavior. The errors observed at the outputs of the system are referred to as D (expected value is one, observed value is zero) and Dbar (expected value is zero, observed value is one).

4.1 AND/OR Fault Trees

The relationship between the structure of a digital circuit (as described above) and the flow of errors through the circuit is easily represented by what we refer to as an AND/OR fault tree. This tree is made up of AND and OR nodes[†] and is derived from the circuit description and fault assumptions. Note that the AND and OR *nodes* as defined are quite distinct from the AND and OR *gates* in the circuit. Figure 1.a depicts an "OR" node. The "OR" node is indicated by the absence of an arc between the outputs of the node. In this figure, if Errorl is observed at the output of component C1, it is either the case that C1 is faulty or the first input equals Valuel OR the second input equals Value2. The variables Valuel and Value2 take on the values 0 or 1, depending on the type of error observed. In Figure 1.b, an "AND" node is depicted. The "AND" node is indicated by an arc between the outputs of the node. In this figure, if Error2 is observed at the output of component C2, it is either the case that C2 is faulty or the first input equals Value3 AND the second input equals Value4. Note that C1 and C2 can be any of the circuit components (e.g., gates or modules). Also, notice that the arrows for the AND/OR fault tree nodes point in the direction opposite to the signal flow direction for the components that the nodes represent. This is to represent the direction of the diagnosis.

Consider the example of an erroneous output from a two input OR gate. If the error observed was a D, it is obvious that either the OR gate was faulty or that both inputs to the gate were zero. Since a one was expected on

[†] There is also a NOT node type. Its meaning will become clear in the examples presented later.

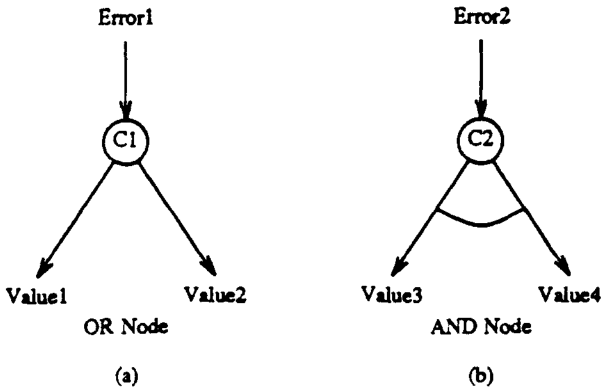

Figure 1: AND/OR Tree Structures

the output, the inputs must have been such that at least one input was a one. The logical representation of this statement is: "the OR gate was faulty" or "input1 = 0" and "input2 = 0." If the OR gate is not faulty, each input which should have value one must actually have value zero. Since the output is a D, there must be at least one input whose value should been one. By considering the possibility of the error propagating through the OR gate, we can now backtrack through the circuit starting with the components that feed the inputs to the OR gate. This relationship between the OR gate and the propagation of a D error through the gate is represented in the AND/OR fault tree structure illustrated in Figure 2. The OR gate in the figure is "named" O1 and has inputs A and B.

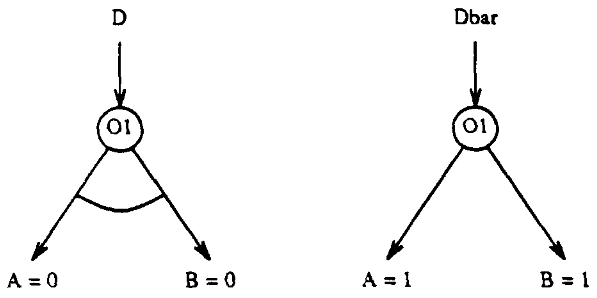

Figure 2: AND/OR Fault Tree for a Faulty OR Gate

If the error was a Dbar on the output of an OR gate, it is obvious that either the OR gate was faulty or that at least one of the inputs to the gate was a one. Since a zero was expected on the output, the inputs should have been such that both were zero. The logical representation of this statement is "the OR gate was faulty" or "input1 = 1" or "input2 = 1." If the OR gate is not faulty, there must exist an input which should have value one but actually has value zero. Since the output is a Dbar, each input should have value zero. Once again, we consider the possibility that the Dbar error propagated through the OR gate by backtracking through the circuit starting with the components which feed the inputs to the OR gate.

Similar techniques are used to represent the flow of errors through other gates. The fault tree structures for other components can be easily obtained by considering the relationship between an error on the output and the the inputs. Figure 3 illustrates the fault tree structures for AND, INVERTER, NOR, and NAND gates. Notice

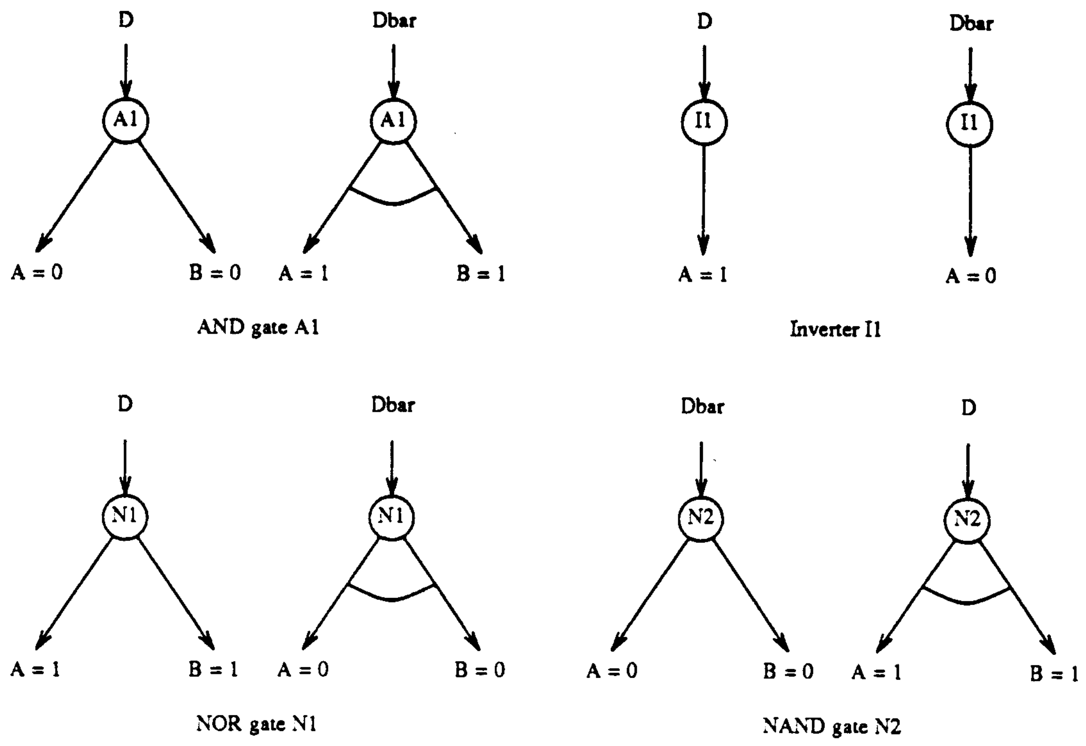

Figure 3: AND/OR Fault Trees for AND, NOT, NAND, and NOR Gates

that the node type for the AND gate is a dual of the node type for the OR gate. Also, the errors for the INVERTER, NOR, and NAND gates are inverted as they pass through the node. This is to be expected, since the outputs from the gates are inverted.

By cascading the appropriate structures for interconnected components, the fault tree for an entire digital system can be obtained. When cascading the tree structures, it is important that the correct structure (either D or Dbar) for a component be used. When an input with the value "input = 0" is to be propagated to another component, a D error component structure is used to feed the input. Similarly, when an input with the value "input = 1" is to be propagated to another component, a Dbar error component structure is used to feed the input.

As an example of this process, the complete D error AND/OR fault tree for the circuit in Figure 4 will be constructed. Assume that a D error on the Z output of OR gate O1 is being considered. The component O1 is represented in the fault tree by the D error node for an OR gate. This fault tree structure can be found in Figure 2. The connection from O1 to A1 is represented in the O1's tree structure by the term "input = 0." As stated above, this means that A1 is represented in the fault tree by the D error node for an AND gate. The connection from O1 to A2 is also represented in O1's tree structure by the term "input = 0." Thus, A2 is represented in the fault tree by the D error node for an AND gate shown in Figure 3. The connection from A1 to A3 is represented in A1's tree structure by the term "input = 0." A3 is then represented in the fault tree by the D error node for an AND

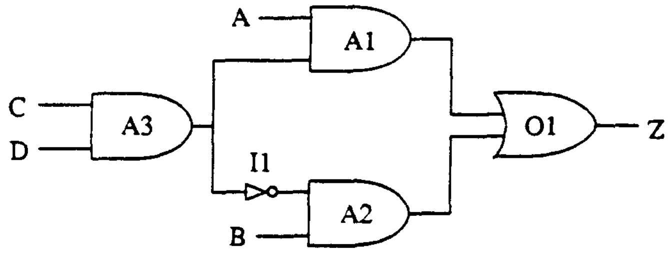

Figure 4: Example Circuit

gate. The connection from A2 to I1 is represented in A2's tree structure by the term "input = 0." Therefore, I1 is represented by the D error node for an INVERTER. The connection from I1 to A3 is represented in I1's tree structure by the term "input = 0." This requires that A3 be represented by a Dbar error structure. Since this is different from the structure representing A3 that had been created previously, another representation of A3 is created. The difference is that the first representation for A3 was a D error structure while the second was a Dbar error structure. This completes the representation of all components in the circuit. The complete AND/OR fault tree for this circuit is shown in Figure 5. Since the Dbar error tree is the dual of the D error tree, it is a simple matter to "invert" the Dbar tree and obtain the D error tree. To invert a tree, simply change all AND nodes to OR nodes (and vice versa) and invert the input values.

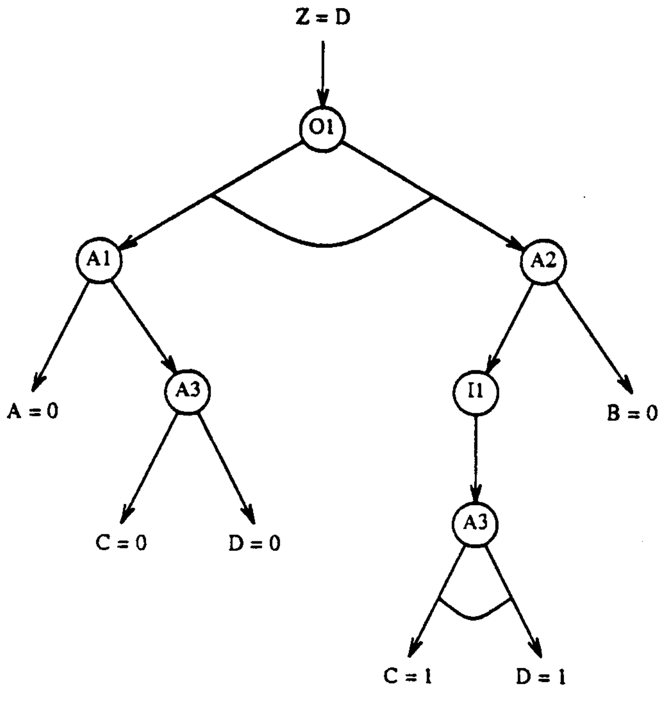

Figure 5: AND/OR Fault Tree for Example Circuit

In the next subsection, the AND/OR fault trees are used to generate clause-form theorems which are used to perform the actual diagnosis. These theorems embody the information contained in the fault tree.

4.2 Clause-Form Theorems

To make it easier to manipulate the information contained in the fault tree representation, the tree is transformed into a clause-form theorem representation. A clause-form theorem is the product-of-sums representation of logical statements. The logical statements describe the relationship between the possible sources of error and the flow of the errors through the structure of the circuit. For the example (in Figure 2) of the OR gate with a D error output, the logical statement to be represented in the theorem is either the gate O1 is faulty or input A equals zero and input B equals zero. In the theorems, the postulate O1' indicates OR gate O1 is faulty while the input postulates A=0 and A=1 indicate that the input A equals zero and one, respectively. The result of transforming the fault tree structure for a D error on the output of OR gate O1 is

```
ThmD1    (O1'  A=0)
ThmD2    (O1'  B=0)
```

In the above representation, there is an implied logical "OR" between each of the statements in a theorem and an implied logical "AND" between each of the theorems. For the example of the OR gate with a Dbar error output, the logical statement to be represented in the theorem is either the gate O1 is faulty or input A equals one or input B equals one. The resulting Dbar theorem is

```
ThmDbar1    (O1'  A=1 B=1)
```

It is trivial to transform the tree structures in Figures 1 and 2 into these theorem representations; for more complex circuits composed of many gates, such a transformation is not as obvious. An algorithm to perform this task is given in Figure 6. In the algorithm, the theorem representation is returned in the variable Theorem_Set.

As an example, we now transform the fault tree representation for the circuit in Figure 4 into a clause-form theorem representation using the algorithm. Given that a D error is observed on the output Z, the function TRAVERSE is applied to the fault tree in Figure 5 and returns the following theorem representation:

```
ThmD1    (O1' A2' I1' A3' B=0 D=1)
ThmD2    (O1' A2' I1' A3' B=0 C=1)
ThmD3    (O1' A1' A3' A=0 C=0 D=0)
```

```
Theorem_Set = TRAVERSE(Output_Node, ∅)

FUNCTION TRAVERSE(Current_Node, Current_Theorems)
    ADD(Current_Node, Current_Theorems)
    IF TYPE(Current_Node) = AND THEN
        New_Theorems = ∅
        FOR EACH CHILD of Current_NODE
            Child_Theorems = TRAVERSE(Child of Current_node, Current_Theorems)
            add Child_Theorems to the list of New_Theorems
        TRAVERSE = New_Theorems
    IF TYPE(Current_Node) = OR THEN
        FOR EACH CHILD of Current_NODE
            Current_Theorems = TRAVERSE(Child of Current_node, Current_Theorems)
        TRAVERSE = Current_Theorems
    IF TYPE(Current_Node) = NOT THEN
        Current_Theorems = TRAVERSE(Child of Current_node, Current_Theorems)
        TRAVERSE = Current_Theorems

PROCEDURE ADD(Current_Node, Current_Theorems)
    IF TYPE(Current-Node) = INPUT THEN
        add the input value for Current_Node to each theorem in Current_Theorems
    ELSE
        add faulty component postulate for Current_Node to each theorem in Current_Theorems

FUNCTION TYPE(Current_Node)
    IF Current-Node is an AND node THEN
        TYPE = AND
    IF Current-Node is an OR node THEN
        TYPE = OR
    IF Current-Node is a NOT node THEN
        TYPE = NOT
    IF Current-Node is a input THEN
        TYPE = INPUT
```

Figure 6: Algorithm to Generate Theorem Representation for an AND/OR Fault Tree

Since the theorems are a sum-of-products representation, when the entire theorem representation is true it must be the case that each individual theorem is true. It then follows that at least one of the postulates in each individual theorem must also be true. Since each theorem contains postulates about both input values and faulty components, the theorems that provide valuable information to the diagnostic process are those with no input postulates. When this is the case, it must be that at least one of the postulates about a faulty component be true. This is where the concept of resolution [25] comes into play. Resolution is a process by which statements known to be false are removed from a set of clause-form theorems.

Suppose that a D error is observed on the Z output of the circuit in figure 4 when the inputs are A=1, B=1, C=1, and D=1. Under nonfaulty conditions, a one should appear on the output. Since a D error is observed, the D error theorem representation must be true. In addition each individual theorem in the representation must be true. Take for example theorems ThmD1 and ThmD2. Since it is already known that C=1 and D=1, nothing can be inferred about the faulty component postulates. In theorem ThmD3, however, we know that the postulates A=0, C=0, and D=0 are false. We can then remove them from the theorem. The theorems that result from the resolution are then used to diagnose the source of the error. A theorem that contains *only* faulty component postulates is referred to as an *activated theorem.* In general, more than one theorem can be activated, but at least one must be. If no theorem is activated, the output could not have been in error for the given inputs. Since the activated theorems are true, one of the faulty component postulates must be true. The components common to all activated theorems are referred to as the suspect set, which is obtained by intersecting the activated theorems. In our example, the only activated theorem is

ThmD3 (O1' A1' A3')

As shown above, the only theorem activated is ThmD3. Its faulty component postulates are O1', A1', A3'; therefore, the suspect set is {O1, A1, A3}.

If new errors are observed (given the same fault) for a different set of inputs, the resolution process can be repeated by generating an additional set of suspects for this new set of input values. In this new diagnosis, the generation of the theorems does not have to be performed again. Assuming that a single fault produced each of the errors, the new set of suspects is intersected with the old set of suspects, resulting in a refined suspect set. This suspect set is no larger than the smallest of the two sets being intersected. Thus, with additional information, the diagnosis may be improved (or, in the worst case, stay the same).

For example, let the input values be changed to A=0, B=1, C=0, D=0. Once again, a D error is observed on the Z output. The known input values are then resolved with the D error theorems to generate the following activated theorems:

ThmD1	(O1' A2' I1' A3')
ThmD2	(O1' A2' I1' A3')

These theorems are intersected to generate the suspect set {O1, A2, I1, A3}. To refine the suspect set, the result of this diagnosis is intersected with the result of the previous diagnosis to generate the new suspect set {O1, A3}. Since the components O1 and A3 are mentioned in each of the theorems, it is impossible to refine the suspect set any further.

In the following chapters, more complex circuits are considered.

5. DIAGNOSTIC REASONING IN CIRCUITS

A more complex example of the diagnosis procedure is illustrated for the circuit depicted in Figure 7. The full fault trees for the C_{n+1} output with D and Dbar errors are shown in Figures 8 and 9, respectively.

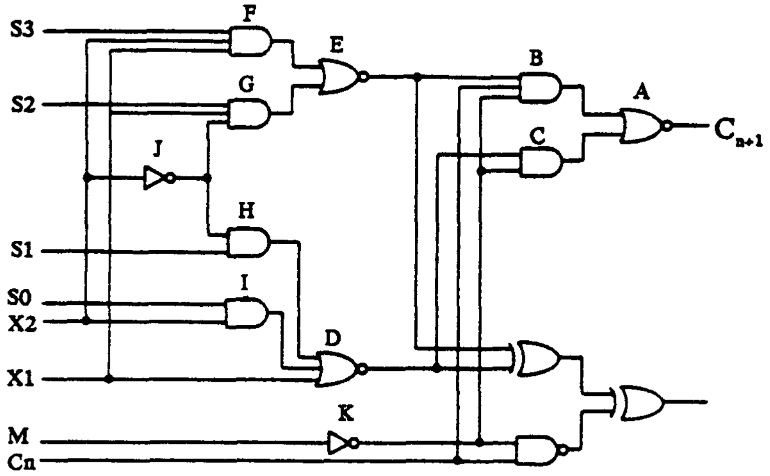

Figure 7: Example Circuit

Tables 1 and 2 contain the theorems generated by traversing the D and Dbar trees for Figure 7. These theorems were generated using the algorithm listed in Figure 6. Note that in the cases of theorems D7, D16, Dbar8, and Dbar12, complementary values for some of the input postulates exist in the same theorem. In the case of theorem D7, the input postulates X2=1 and X2=0 are both present. Since only one of these can be resolved with a known input value, the theorem can never be activated. Therefore, the theorems D7, D16, Dbar8 and Dbar12 are discarded.

Consider a D error observed when the known inputs were Cn=0 M=1 S0=0 S1=1 S2=0 S3=0 X1=1 X2=1. If these input values are resolved with the D theorems, four theorems are activated. The activated theorems are

ThmD1	(A' B' C' K')
ThmD2	(A' B' C' K')
ThmD9	(A' B' C' D' K')
ThmD10	(A' B' C' D')

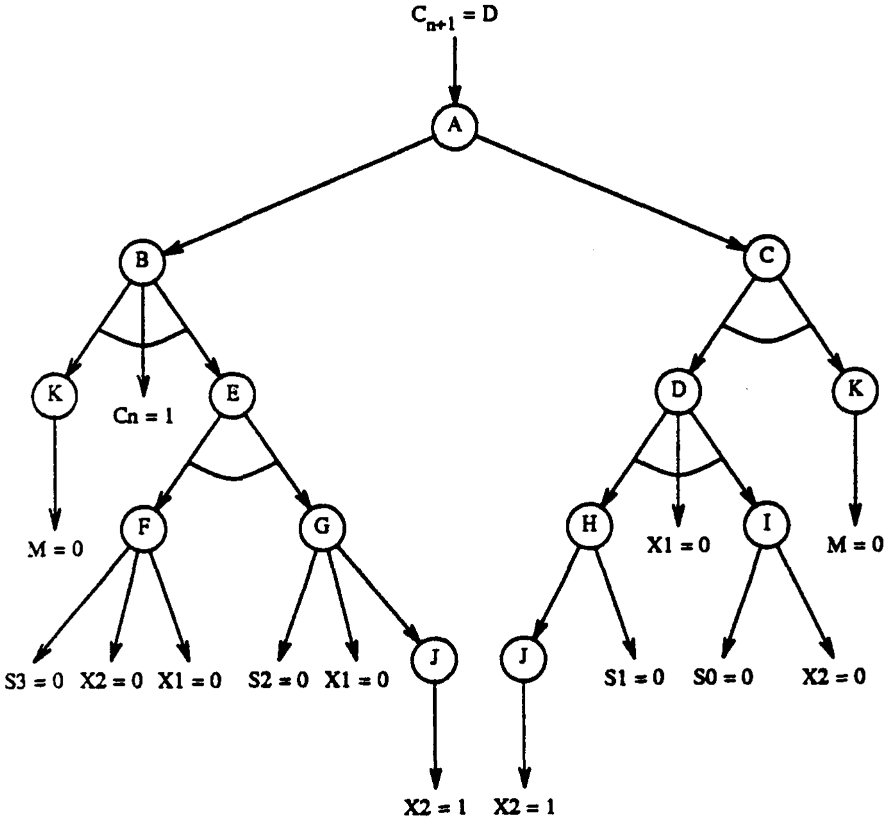

Figure 8: Fault Tree for D Error for Circuit in Figure 7

Table 1 : Theorems for D Error for Circuit in Figure 7

Theorem #	Theorem	
D1	(A' B' C' K'	M=0)
D2	(A' B' C' K'	Cn=1 M=0)
D3	(A' B' C' E' F' K'	M=0 S3=0 X1=0 X2=1)
D4	(A' B' C' E' G' J' K'	M=0 S2=0 X1=0 X2=1)
D5	(A' B' C' D' H' J' K'	M=0 S1=0 X2=1)
D6	(A' B' C' D' H' J'	Cn=1 S1=0 X2=1)
D7	(A' B' C' D' E' F' H' J'	S1=0 S3=0 X1=0 X2=0 X2=1)
D8	(A' B' C' D' E' G' H' J'	S1=0 S2=0 X1=0 X2=1)
D9	(A' B' C' D' K'	M=0 X1=0)
D10	(A' B' C' D'	Cn=1 X1=0)
D11	(A' B' C' D' E' F'	S3=0 X1=0 X2=0)
D12	(A' B' C' D' E' G' J'	S2=0 X1=0 X2=0)
D13	(A' B' C' D' I' K'	M=0 S0=0 X2=0)
D14	(A' B' C' D' I'	Cn=1 S0=0 X2=0)
D15	(A' B' C' D' E' F' I'	S0=0 S3=0 X1=0 X2=0)
D16	(A' B' C' D' E' G' I' J'	S0=0 S2=0 X1=0 X2=0 X2=1)

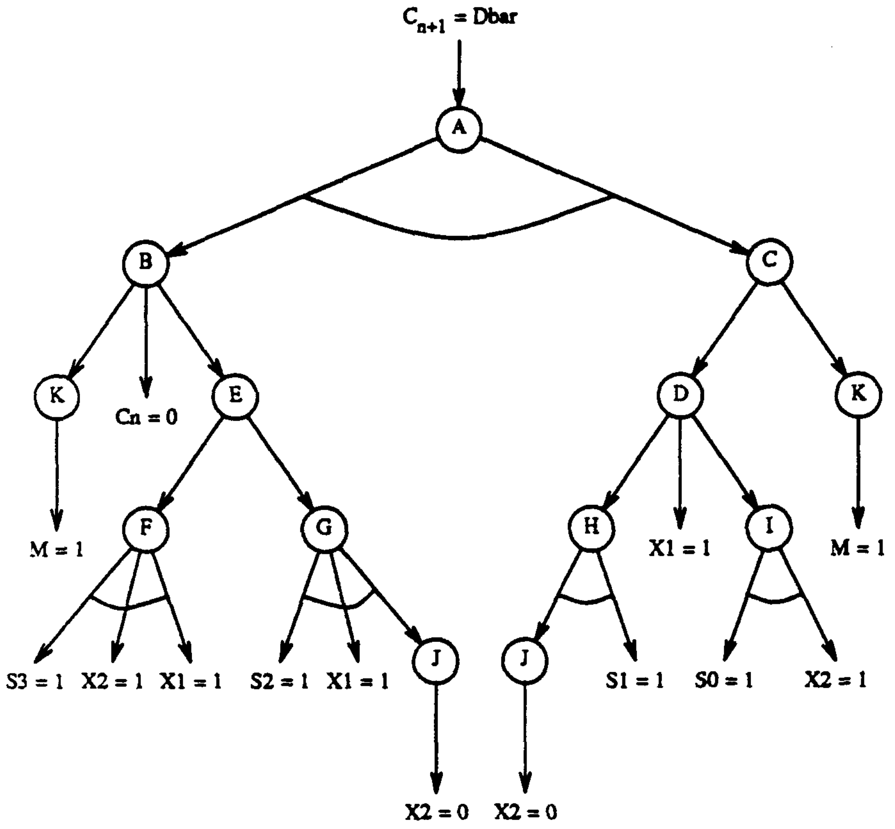

Figure 9: Fault Tree for Dbar Error for Circuit in Figure 7

Table 2 : Theorems for Dbar Error for Circuit in Figure 7

Theorem #	Theorem
Dbar1	(A' B' E' F' G' K' Cn=0 M=1 S2=1 S3=1)
Dbar2	(A' B' E' F' G' K' Cn=0 M=1 S2=1 X2=1)
Dbar3	(A' B' E' F' G' K' Cn=0 M=1 S2=1 X1=1)
Dbar4	(A' B' E' F' G' K' Cn=0 M=1 S3=1 X1=1)
Dbar5	(A' B' E' F' G' K' Cn=0 M=1 X1=1 X2=1)
Dbar6	(A' B' E' F' G' K' Cn=0 M=1 X1=1)
Dbar7	(A' B' E' F' G' J' K' Cn=0 M=1 S3=1 X2=0)
Dbar8	(A' B' E' F' G' J' K' Cn=0 M=1 X2=0 X2=1)
Dbar9	(A' B' E' F' G' J' K' Cn=0 M=1 X1=1 X2=0)
Dbar10	(A' C' D' H' I' J' K' M=1 S0=1 X1=1 X2=0)
Dbar11	(A' C' D' H' I' K' M=1 S0=1 S1=1 X1=1)
Dbar12	(A' C' D' H' I' J' K' M=1 X1=1 X2=0 X2=1)
Dbar13	(A' C' D' H' I' K' M=1 S1=1 X1=1 X2=1)

Intersecting these theorems results in the suspect set {A, B, C}. Thus, one of the components A, B, or C is faulty.

There are two approaches to the refinement of the suspect set. In the first approach, a refinement is performed when the next error is observed. It has been shown that the error recurrence rate due to a fault in a real system is quite high [26]. Given this, it is highly likely that a new set of inputs which can be used to refine the diagnosis will be available within a short period of time. For example, assume that the inputs M and X1 both change from one to zero and a Dbar error is observed on the output of C_{n+1}. When these new known input values are resolved with the Dbar theorems, only one is activated:

ThmDbar10 (A' C' D' H' I' J' K')

The suspect set for this Dbar error is {A, C, D, H, I, J, K}. The suspect set from the first observed error ({A, B, C}) is now intersected with the suspect set from the second observed error to produce the refined suspect set {A, C}. Thus, the suspects have been reduced to two possibilities: the NOR gate A or the AND gate C.

The second approach to suspect refinement requires the generation of specific input values which can differentiate between suspects. For example, since theorem Dbar1 does not contain a faulty component postulate for component C, a set of known inputs that activate theorem Dbar1 could be applied to determine if C is the faulty component. If with the inputs Cn=1 M=0 S2=0 S3=0 (all other inputs are don't cares) a Dbar error is observed, the new suspect set will not contain C. Thus, an intersection of the old suspect set and the new suspect set will remove C from suspicion. The resultant suspect set will contain only A. If this input pattern does not produce an error, the diagnosis cannot be further refined using that set of known inputs. Since a postulate about component A is included in every theorem (for both D and Dbar errors), it is impossible to remove the output of A from the suspect set. This would seem reasonable since the output of A is the output of the circuit being diagnosed. Whenever an error is observed, it is possible that the output of A produced it. This problem of determining input patterns that distinguish between faulty components reduces to the covering problem. Similar to test pattern generation, this problem has been shown to be NP-Complete [27]. Heuristic methods, such as AQ [28], can be used to help solve this problem.

It is noted that much of the work done in resolving and intersecting the theorems can be done in advance, when the time constraints are not critical. For all known input combinations, the D and Dbar suspect sets can be generated in the form of a table (referred to as a *single step suspect table*). Given an error, the table can be used to generate the suspect set for the input combination. Thus, each diagnosis can be performed in a single step. For example, consider the theorems for a Dbar error listed in Table 2. The theorems are transformed into the single step suspect table listed below:

CnS0S1S2S3X1X2	S	CnS0S1S2S3X1X2	S	CnS0S1S2S3X1X2	S	CnS0S1S2S3X1X2	S
1111111	-	1011111	-	0111111	-	0011111	-
1111110	-	1011110	-	0111110	-	0011110	-
1111101	α	1011101	γ	0111101	-	0011101	ε
1111100	α	1011100	α	0111100	-	0011100	-
1111011	β	1011011	β	0111011	-	0011011	-
1111010	-	1011010	-	0111010	-	0011010	-
1111001	α	1011001	γ	0111001	-	0011001	ε
1111000	α	1011000	α	0111000	-	0011000	-
1110111	-	1010111	-	0110111	-	0010111	-
1110110	α	1010110	α	0110110	-	0010110	-
1110101	α	1010101	γ	0110101	-	0010101	ε
1110100	α	1010100	α	0110100	-	0010100	-
1110011	α	1010011	α	0110011	-	0010011	-
1110010	α	1010010	α	0110010	-	0010010	-
1110001	α	1010001	γ	0110001	-	0010001	ε
1110000	α	1010000	α	0110000	-	0010000	-
1101111	-	1001111	-	0101111	-	0001111	-
1101110	-	1001110	-	0101110	-	0001110	-
1101101	α	1001101	γ	0101101	-	0001101	ε
1101100	γ	1001100	γ	0101100	δ	0001100	δ
1101011	β	1001011	β	0101011	-	0001011	-
1101010	-	1001010	-	0101010	-	0001010	-
1101001	α	1001001	γ	0101001	-	0001001	ε
1101000	γ	1001000	γ	0101000	δ	0001000	δ
1100111	-	1000111	-	0100111	-	0000111	-
1100110	α	1000110	α	0100110	-	0000110	-
1100101	α	1000101	γ	0100101	-	0000101	ε
1100100	γ	1000100	γ	0100100	δ	0000100	δ
1100011	α	1000011	α	0100011	-	0000011	-
1100010	α	1000010	α	0100010	-	0000010	-
1100001	α	1000001	γ	0100001	-	0000001	ε
1100000	γ	1000000	γ	0100000	δ	0000000	δ

The five suspect classes are as follows: α = {A,B,E,F,G,K}, β = {A,B,E,F,G,J,K}, γ = {A,K}, δ = {A,C,D,H,I,K}, and ε = {A,C,D,H,I,J,K}. Notice that the input M was not included in the table. Since it is necessary that M be zero for every Dbar error, to including M in the table will not differentiate between any of the suspects. If the input values hash onto an empty suspect class (indicated in the table by a "-"), the known input values could not have produced a Dbar error on the C_{n+1} output. In the best case, the size of the table will be equal the number of different suspect classes generated by applying the input values which can produce an error. In the worst case, the table will have 2^n entries, where n is the number of inputs to the circuit being diagnosed. Techniques such as Quine-McCluskey minimization can be used to reduce the size of the table. Let us return to the example diagnosis done earlier. For the inputs Cn=0 M=0 S0=0 S1=1 S2=0 S3=0 X1=0 X2=1, a Dbar error was observed. Using these inputs as a table address, the suspect class ε is obtained. As expected, the suspect set for this set of input values is {A,C,D,H,I,J,K}.

The question arises: How applicable is this method to large circuits? The algorithm in Figure 6 to generate the theorem representation can be easily automated. The generation of a single step suspect table can also be automated. Thus, automating the diagnosis methodology to handle large circuits is a realistic goal. An important problem that still remains is an exponential growth in the search complexity (i.e., the size of the table) as the circuit size increases. A solution to this problem is discussed in the following chapter.

6. PARTITIONING TO REDUCE DIAGNOSIS COMPLEXITY

The worst case complexity of performing a diagnosis (in terms of the number of theorems) can grow exponentially with the size of the circuit. For large circuits, this growth is unacceptable. To reduce the complexity, now introduced is a method of partitioning the circuit under diagnosis which considerably reduces the number of theorems that need to be generated. This is due to the fact that each partition is diagnosed separately, and as a result the number of theorems, although still exponential in the number of inputs to each partition, grows linearly with the number of partitions. When a runtime diagnosis is performed, the partitions which provide no information to the diagnosis are pruned from the search space. The algorithm to diagnose the partitioned circuit is as follows:

Step 1: Generate the theorems for each partition. On observing an error, start at the partition where the error is observed.

Step 2: For the current partition, resolve the known input values with the appropriate set of error theorems.

Step 3: If any of the theorems are fully[†] activated, intersect these theorems and return the result.

Step 4: If there are no fully activated theorems, collect the partially[††] activated theorems. In this case, it is still possible to activate some of the partially activated theorems, but they need to consider the partitions which output the unresolved inputs. For each unresolved input which is connected to the output of another partition, recursively apply Steps 2 - 4 on the appropriate partition output. An unresolved input postulate is replaced by the faulty component postulates returned by the recursive application of Steps 2 - 4. After all partially activated theorems are processed, intersect the partially activated theorems and return the result.

As an example of this process, consider the N-bit carry-ripple adder in Figure 10. It is partitioned into N partitions, where each partition is a full adder (FA_0 - FA_{N-1}). The problem is to determine which full adder is

[†] A fully activated theorem is one in which there are only faulty component postulates.

[††] A partially activated theorem is one in which the only unresolved input postulates are those connected to outputs of other partitions.

faulty and within the full adder, which gate is faulty. The circuit representing the internal structure of each of the full adders is illustrated in Figure 11. Figures 12 - 15 show the AND/OR fault trees for D and Dbar errors on the sum and carry outputs of a full adder module. Tables 4 - 7 list the theorems generated from the fault tree structures.

We will now diagnose a 4-bit carry-ripple adder (i.e., N equals 4) using partitioning. The components are labeled FA$_i$:K where FA$_i$ is full adder i and K identifies the specific component within FA$_i$. For example, FA$_3$:O1 identifies the OR gate O1 within full adder 3. Assume that the initial carry input C_0 and all inputs A_i and B_i are set to zero. This should produce a zero on the sum output S_3. Instead, a Dbar error is observed. We first apply the known input values to the partition producing the erroneous output. The result is the Dbar theorems for S_3 (from Table 5). On resolving the input values, A_3=0 and B_3=0 produce the following:

(S_3) Dbar1	(FA$_3$:O2’ FA$_3$:A4’ FA$_3$:A5’ FA$_3$:A6’ FA$_3$:A7’ C_3=1)
(S_3) Dbar2	(FA$_3$:O2’ FA$_3$:A4’ FA$_3$:A5’ FA$_3$:A6’ FA$_3$:A7’ FA$_3$:I4’ FA$_3$:I5’ B_3=0 C_3=0)
(S_3) Dbar3	(FA$_3$:O2’ FA$_3$:A4’ FA$_3$:A5’ FA$_3$:A6’ FA$_3$:A7’ FA$_3$:I1’ FA$_3$:I6’ A_3=0 C_3=0)
(S_3) Dbar4	(FA$_3$:O2’ FA$_3$:A4’ FA$_3$:A5’ FA$_3$:A6’ FA$_3$:A7’ FA$_3$:I2’ FA$_3$:I3’ A_3=0 B_3=0 C_3=1)

As can be seen, there are no fully activated theorems. In addition, there is only one partially activated theorem (Dbar1). Here, the unresolved input postulate connected to another partition is C_3=1. It is connected to the carry output of full adder FA$_2$. Now the Dbar theorems for the carry output of FA$_2$ are resolved with the known inputs. The result is

(C_3) Dbar1	(FA$_2$:O1’ FA$_2$:A1’ FA$_2$:A2’ FA$_2$:A3’)
(C_3) Dbar2	(FA$_2$:O1’ FA$_2$:A1’ FA$_2$:A2’ FA$_2$:A3’ C_2=1)
(C_3) Dbar3	(FA$_2$:O1’ FA$_2$:A1’ FA$_2$:A2’ FA$_2$:A3’ C_2=1)
(C_3) Dbar4	(FA$_2$:O1’ FA$_2$:A1’ FA$_2$:A2’ FA$_2$:A3’ C_2=1)

In this case, there is a single fully activated theorem. It is Dbar1. This theorem is returned to the Dbar1 theorem from partition FA$_3$ and replaces the input postulate C_3=1. The original partially activated theorem becomes

(S_3) Dbar1 (FA$_3$:O2’ FA$_3$:A4’ FA$_3$:A5’ FA$_3$:A6’ FA$_3$:A7’ FA$_2$:O1’ FA$_2$:A1’ FA$_2$:A2’ FA$_2$:A3’)

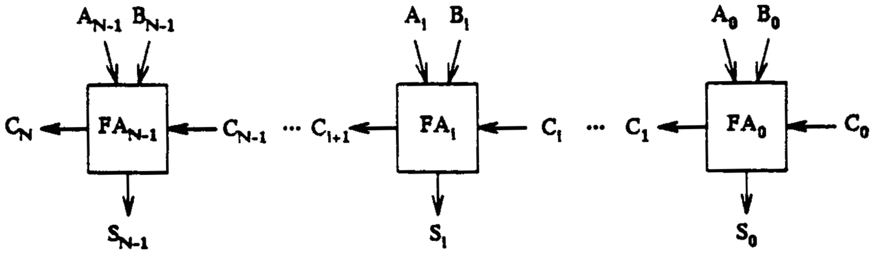

Figure 10: Block Diagram for an N-Bit Carry-Ripple Adder

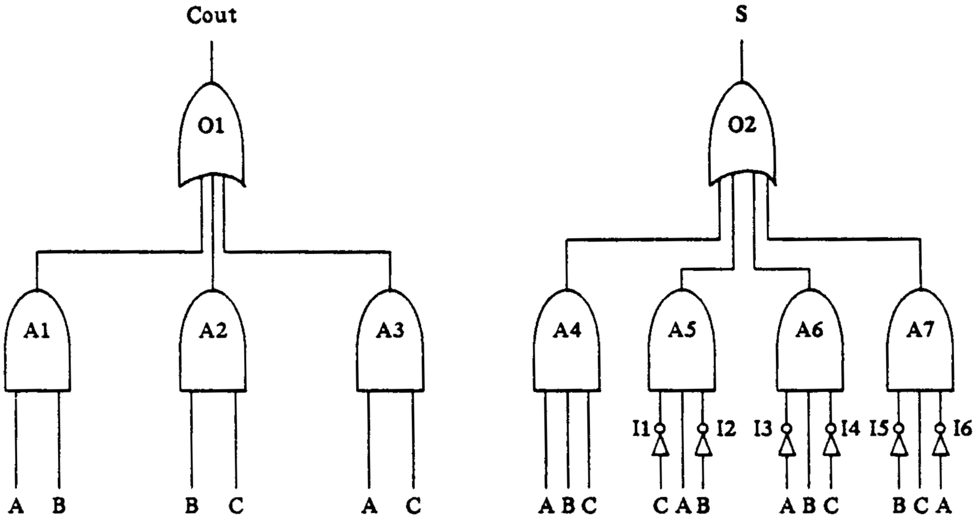

Figure 11: Gate Level Circuit Diagram for Full Adder

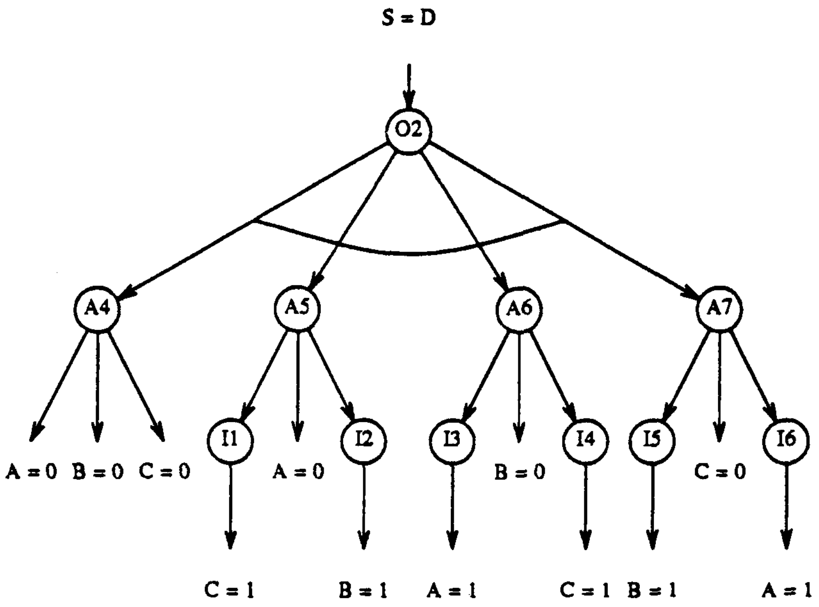

Figure 12: Fault Tree for D Error for S Output of Full Adder

Table 4: Theorems for D Error on S Output

Theorem #	Theorem	
D1	(O2' A4'	A=0 B=0 C=0)
D2	(O2' A5' I1' I2'	A=0 B=1 C=1)
D3	(O2' A6' I3' I4'	A=1 B=0 C=1)
D4	(O2' A7' I5' I6'	A=1 B=1 C=0)

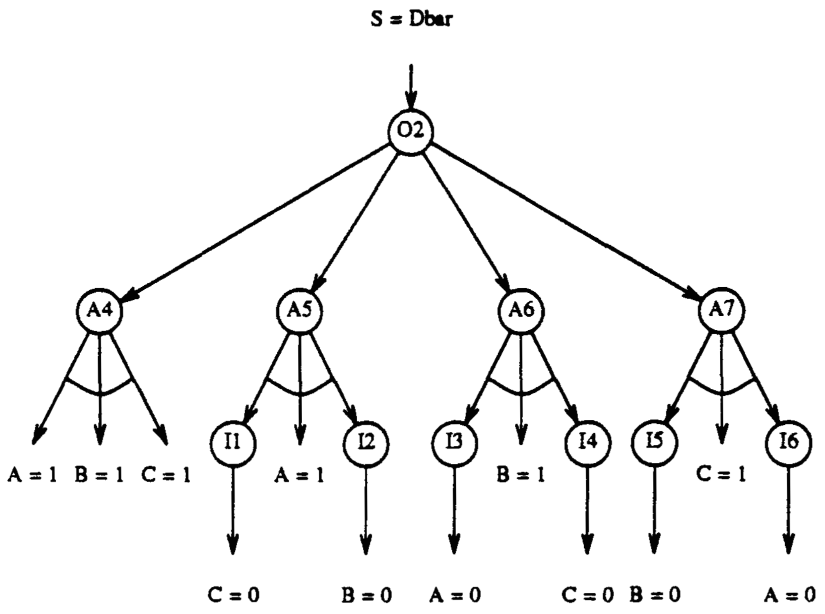

Figure 13: Fault Tree for Dbar Error for S Output of Full Adder

Table 5: Theorems for Dbar Error on S Output

Theorem #	Theorem
Dbar1	(O2’ A4’ A5’ A6’ A7’ A=1 B=1 C=1)
Dbar2	(O2’ A4’ A5’ A6’ A7’ I4’ I5’ A=1 B=0 C=0)
Dbar3	(O2’ A4’ A5’ A6’ A7’ I1’ I6’ A=0 B=1 C=0)
Dbar4	(O2’ A4’ A5’ A6’ A7’ I2’ I3’ A=0 B=0 C=1)

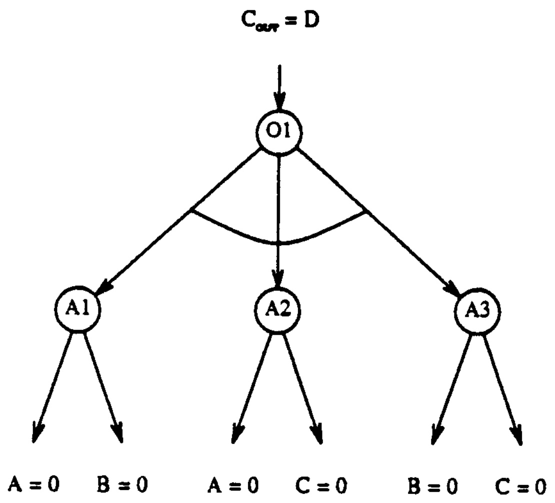

Figure 14: Fault Tree for D error on C_{OUT} Output

Table 6: Theorems for D Error on C_{OUT} Output

Theorem #	Theorem
D1	(O1' A1' A=0 B=0)
D2	(O1' A2' A=0 C=0)
D3	(O1' A3' B=0 C=0)

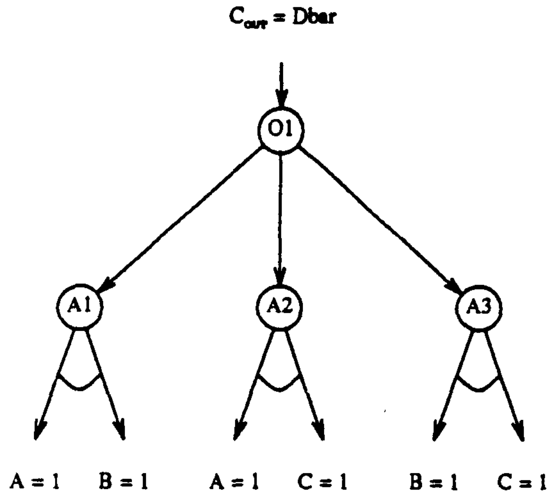

Figure 15: Fault Tree for Dbar error on C_{OUT} Output

Table 7: Theorems for Dbar Error on C_{OUT} Output

Theorem #	Theorem
Dbar1	(O1’ A1’ A2’ A3’ A=1 B=1)
Dbar2	(O1’ A1’ A2’ A3’ A=1 C=1)
Dbar3	(O1’ A1’ A2’ A3’ B=1 C=1)
Dbar4	(O1’ A1’ A2’ A3’ A=1 B=1 C=1)

Thus, for the observed error value and known inputs, the complete suspect set is { FA$_3$:O2 FA$_3$:A4 FA$_3$:A5 FA$_3$:A6 FA$_3$:A7 FA$_2$:O1 FA$_2$:A1 FA$_2$:A2 FA$_2$:A3 }. Note that the suspect set has been reduced to eight of the sixty components in the carry ripple adder.

Now assume that another error caused by the same fault is observed for the inputs $A_3 = 0$, $B_3 = A_2 = B_2 = A_1 = B_1 = A_0 = B_0 = 1$. The error observed is a D on the carry output of FA$_3$. We then apply the known input values to the partition containing the erroneous output. These are the D theorems for C_4 (from Table 6). The following are the result of resolving the values A_3=0 and B_3=1 with the theorems:

(C$_4$) D1 (FA$_3$:O1’ FA$_3$:A1’ A$_3$=0)
(C$_4$) D2 (FA$_3$:O1’ FA$_3$:A2’ A$_3$=0 C$_3$=0)
(C$_4$) D3 (FA$_3$:O1’ FA$_3$:A3’ C$_3$=0)

As can be seen, there are no fully activated theorems. There is only one partially activated theorem (D3). Here, the unresolved input postulate connected to another partition is C_3=0. It is connected to the carry output of full adder FA$_2$. Now the D theorems for the carry output of FA$_2$ are resolved with the known inputs. The result is

(C$_3$) D1 (FA$_2$:O1’ FA$_2$:A1’)
(C$_3$) D2 (FA$_2$:O1’ FA$_2$:A2’ C$_2$=0)
(C$_3$) D3 (FA$_2$:O1’ FA$_2$:A3’ C$_2$=0)

The above set has a single fully activated theorem. It is D1. This theorem is returned to the D1 theorem from partition FA$_3$ and replaces the input postulate C_3=0. The original partially activated theorem becomes

(C$_4$) D3 (FA$_3$:O1’ FA$_3$:A3’ FA$_2$:O1’ FA$_2$:A1’)

Thus, for this observed error value and known input values, the suspect set is { FA$_3$:O1 FA$_3$:A3 FA$_2$:O1 FA$_2$:A1 }. Intersecting this suspect set with the suspect set from the first diagnosis generates the refined suspect set { FA$_2$:O1 FA$_2$:A1 }. The suspect now has been reduced to two (out of sixty) components: AND gate A1 in FA$_2$ or the OR gate O1 in FA$_2$.

6.1 Discussion

If the entire N-bit adder were considered as a single partition, the number of theorems would grow exponentially with the number of bits. This can be verified by generating a complete fault tree for the N-bit adder and traversing the tree using the algorithm in Figure 6. By partitioning the circuit, the total number of theorems grows linearly with the number of bits. This too can be verified by applying the algorithm in Figure 6 to each partition and summing the total number of theorems generated. Table 3 illustrates the differences between the total number of theorems generated with and without partitioning. In the worst case, the diagnosis will have to examine each theorem. As can be easily seen, the worst case complexity is drastically reduced by partitioning.

Table 3: Complexity of Diagnosis with and without Partitioning

Error	Number of Theorems	
	Without Partitioning	With Partitioning
D on Sum	$2 \times 3^{N-1} + 2 \times 4^{N-1}$	7N - 3
D on Carry	3^N	3N
Dbar on Sum	$2 \times 3^{N-1} + 2 \times 4^{N-1}$	7N - 3
Dbar on Carry	4^N	4N

The single step suspect table introduced in section 5 can also be used with a partitioned circuit. Each partition has a single step suspect set table associated with it and by using partitioning the runtime complexity of the diagnosis will now be linear in the number of *partitions*. Of course, if each component is set to be an individual partition, the time complexity of the diagnosis will be linear in the number of components. Thus, by first partitioning the circuit and then generating a single step suspect table for each partition, the runtime complexity of a diagnosis has been drastically reduced. Instead of a runtime complexity linear in the number of components, it is now linear in the number of partitions. By choosing the appropriate partitioning, a reasonable tradeoff between the space complexity (i.e., the size of the single step suspect table) and the time complexity required to perform a diagnosis can be achieved. In comparison, the work presented in [23] (which in essence considers each component to be a partition) has an exponential time complexity[†].

[†] Although not explicitly stated in [23], if the algorithm is modified to save some information (thus increasing the time and space complexity of the diagnosis), the worst case time complexity can be improved to be linear in the number of components.

7. DIAGNOSIS AT THE MODULE LEVEL

It is often the case that the diagnosis need not be performed down to the gate level. If the replaceable components are at the module level, it is inefficient to obtain a diagnosis at a level deeper than the module. Besides providing more information than necessary, a gate level diagnosis would take more time than a module level diagnosis. The techniques developed to diagnose at the gate level can be modified to allow a diagnosis at a higher level. In performing a diagnosis at the module level, a module is first described by its logical function in the form of truth table or Karnaugh map. This function is then transformed into a product-of-sums representation (i.e., a theorem representation). A faulty component postulate for the module being considered is added to each sum clause in this representation. This theorem representation can then be used as nodes in a fault tree at the module level. As with the gate level diagnosis, the module level circuit can be partitioned to reduce the complexity of the diagnosis. As an example, consider a module M described by the truth table in Table 8.

Table 8: Truth Table for Module M

Inputs				Output
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

groups of modules, the partitioning technique is recursively applied first to each partition and then to the modules within the partitions.

8. CONCLUSIONS

In this thesis a methodology to perform the diagnosis digital systems was introduced. The methodology, based on the generation of an AND/OR fault tree which is traversed to generate clause-form theorems, uses resolution techniques to determine the conditions necessary to activate faulty components. If additional errors occur due to the same fault, the set of suspects is refined by intersecting the new suspect sets with the previous suspect set. More importantly, a method of partitioning the circuit was introduced to reduce the complexity of the diagnosis. By using a combination of partitioning and the single step suspect table, it has been shown that a runtime diagnosis has a time complexity linear in the number of partitions.

8.1 Future Research

Future work in the area of diagnostic reasoning for digital systems should include techniques to take into consideration more complex fault models such as the stuck open model which introduces sequential behavior into the circuit. In addition, it is not always reasonable to assume that there is only a single fault present in the system. If the mean-time-to-repair is greater than the mean-time-to-failure for components in the system, it is entirely possible that two faults are present in the system simultaneously. This complicates the diagnostic process since faults could possibly mask each other depending on the inputs to the system.

The greatest gains in the actual implementations will probably be found in hybrid diagnostic systems which use a combination of rule-based techniques and reasoning approaches. The rule-based techniques could be first used to prune the set of suspect components. Once that has been accomplished, a reasoning approach could be used to further refine the set of suspects.

REFERENCES

[1] F.P. Preparata, G. Metze, and R.T. Chien, "On the Connection Assignment Problem of Diagnosable Systems," *IEEE Transactions on Electronic Computers*, pp. 843-854, Dec. 1967.

[2] A.T. Dahbura and G.M. Masson, "An O($n^{2.5}$) Fault Identification Algorithm for Diagnosable Systems," *IEEE Transactions on Computers*, pp. 486-492, June 1984.

[3] B. Chandrasekaran and R. Milne, Eds., "Special Section on Reasoning about Structure, Behavior and Function," *SIGART Newsletter*, pp. 4-55, July 1985.

[4] J. De Kleer, "How Circuits Work," *Artificial Intelligence*, pp. 205-280, Dec. 1984.

[5] B. Kuipers, "Commonsense Reasoning about Causality: Deriving Behavior from Structure," *Artificial Intelligence*, pp. 169-203, Dec. 1984.

[6] R.T. Hartley, "CRIB: Computer Fault-Finding through Knowledge Engineering," *IEEE Computer*, pp. 76-83, Mar. 1984.

[7] F. Pipitone, "An Expert System for Electronics Troubleshooting Based on Function and Connectivity," *Proceedings of the Second Conference on Artificial Intelligence Applications*, pp. 34-41, Dec. 1983.

[8] F. Pipitone, "The FIS Electronics Troubleshooting System," *IEEE Computer*, pp. 68-76, July 1986.

[9] M.P. Prevost and T.J. Laffey, "Knowledge-Based Diagnosis of Electronic Instrumentation," *Proceedings of the Second Conference on Artificial Intelligence Applications*, pp. 42-48, Dec. 1983.

[10] M. Ragheb and D. Gvillo, "Development of Model-Based Fault-Identification Systems on Microcomputers," SPIE Vol. 635: *Applications of Artificial Intelligence III*, 1986, pp. 268-275.

[11] M. Ragheb, D. Gvillo, and H. Makowitz, "Symbolic Simulation of Engineering Systems on a Supercomputer," SPIE Vol. 635: *Applications of Artificial Intelligence III*, 1986, pp. 368-374.

[12] G.R. Gottschalk and R.M. Vandoorn, "A Rule-Based System to Diagnose Malfunctioning Computer Peripherals," *Hewlett-Packard Journal*, pp. 48-53, Nov. 1986.

[13] H. Shubin and J.W. Wade, "IDT: An Intelligent Diagnostic Tool," *Proceedings of AAAI 1982*, pp. 290-295, Aug. 1982.

[14] C. Strandberg, I. Abramovich, D. Mitchell, and K. Prill, "PAGE-1: A Troubleshooting Aid for Nonimpact Page Printing Systems, *Proceedings of the Second Conference on Artificial Intelligence Applications*, pp. 68-74, Dec. 1983.

[15] A.J. Wilkinson, "MIND: An Inside Look at and Expert System for Electronic Diagnosis," *IEEE Design and Test*, pp. 69-77, Aug. 1985.

[16] N. Yamada and H. Motoda, "A Diagnosis Method of Dynamic System Using the Knowledge on System Description," *Proceedings of IJCAI-8*, pp. 225-229, Aug. 1983.

[17] R. Davis and H. Shobe., "Representing Structure and Behavior of Digital Hardware," *IEEE Computer*, pp. 75-82, Oct. 1983.

[18] R. Davis, H. Shrobe, W. Hamshcher, K. Wieckert, M. Shirley, and S. Polit, "Diagnosis Based on Description of Structure and Function," *Proceedings of AAAI 1982*, pp. 137-142, Aug. 1982.

[19] R. Davis, "Diagnosis Via Causal Reasoning: Paths of Interaction and the Locality Principle," *Proceedings of AAAI 1983*, pp. 88-99, Aug. 1983.

[20] R. Davis, "Diagnostic Reasoning Based on Structure and Behavior," *Artificial Intelligence*, pp. 347-410, Dec. 1984.

[21] R. Davis, "Reasoning from First Principles in Electronic Troubleshooting," *International Journal of Man-Machine Studies*, pp. 403-423, Nov. 1983.

[22] M. Genesereth, "Diagnosis Using Hierarchical Design Models," *Proceedings of AAAI 1982*, pp. 278-283, Aug. 1982.

[23] M. Genesereth, "The Use of Design Descriptions in Automated Diagnosis," *Artificial Intelligence*, pp. 411-436, Dec. 1984.

[24] R.E. Barlow and H.E. Lambert, "Introduction to Fault Tree Analysis," in *Reliability and Fault Tree Analysis*. Philadelphia: SIAM, 1975, pp. 7-35.

[25] J.A. Robinson, "A Machine-Oriented Logic Based on the Resolution Principle," *Journal of the ACM*, pp. 23-41, Jan. 1965.

[26] R.K. Iyer, D.J. Rossetti, and M.C. Hsueh, "Measurement and Modeling of Computer Reliability as Affected by System Activity," *ACM Transactions on Computer Systems*, pp. 214-237, Aug. 1986.

[27] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[28] R.S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The AQ15 Learning System: An Overview and Experiments," Report No. UIUCDCS-R-86-1260, University of Illinois at Urbana-Champaign, Department of Computer Science, July 1986.